Operation of pDAQ for RPC Factory for Muon Trigger Upgrade

Daniel Jumper, Amanda Caringi

August 20, 2007

1. Introduction

For the muon trigger upgrade resistive plate chambers (RPCs) will play an integral part. These RPCs will be assembled into larger sections for installation and tested for quality in the "RPC factory" in building 912. This note concerns the data acquisition (DAQ) hardware and software used in the cosmic ray test stand of RPC factory.

Initial components of the DAQ are already in use and have been used for preliminary efficiency and timing tests on scintillator paddles. The operation of the DAQ and descriptions of these tests, as the operation is concerned, are described below. Very basic descriptions are included for those not familiar with the DAQ system.

2. Hardware

2.1 *CAMAC crates*

CAMAC crates house the data taking modules (TDCs and ADCs) and other utility modules. The RPC factory currently has three CAMAC crates. Our tests were done using only one crate.

- *How to turn on and off*

  The important thing to know about powering the CAMAC crate on and off is the controller module. To power up the crate you must turn the power on the crate first, then you turn the controller online. To power down the crate you must first turn the controller offline, then you turn the power to the crate off. Once you recycle power on a crate, you may notice new indicator lights turn on on the controller module, this is usually fixed by rebooting the VME computer. It is good practice to reboot the VME regardless when recycling power on the CAMAC crate.
- *How to install modules*

  The power to the CAMAC crate must be off to install or remove modules. Once the power is off, screw in or unscrew the module as needed. The modules need to be initialized on the software end, please see section 3 for further instruction.

2.2 *NIM*

The NIM crates house and provide power for functional modules. No outside interface is required.

- *How to turn on and off*

  Recycling power is trivial; you can just turn it on and off
- *How to install modules*

  As with the CAMAC crate, the power must be off before you install or remove a module.

2.3 *VME*

The VME crate is more self contained and houses the mini-computer that commands the CAMAC and interfaces with the computer; it is the link between the crates and the computer. For more details on computer interface see section 2.6.

- *Connection to CAMAC*

  The CAMAC branch highway serial cable connects from the VME card to the CAMAC crate controller module. The VME also needs to receive a trigger signal. The trigger signal should plug into the 'init4' plug of the VME card. This trigger tells the system when to take data for an event.

2.4 *Computer* (see section 3 for usage)

- Username: *pdaq*
- Password: *phenix%rpc*

2.5 *Test stand / Specific setup*

- *Test stand setup*

  The test stand used to test for scintillator efficiency sets up three scintillators stacked. Each scintillator is attached to a layer of foam core for stability. The rack has a series of four pegs on three levels where the scintillators rest. A small scintillator is used as part of the trigger logic. The small scintillator can be easily moved around the setup according to your needs. Changing the setup for the scintillators is an easy process of sliding the old scintillator our and sliding the new one in. A schematic of the stand is seen in Figure 1 with each photo multiplier tube (PMT) labeled. Figure 2 shows a photo of the setup also with the PMTs labeled.

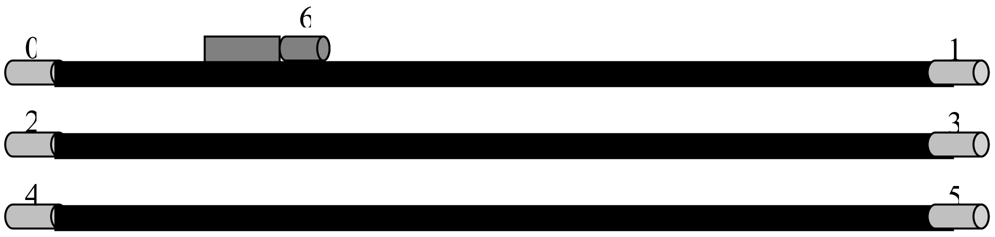

Figure 1: Schematic of scintillator test stand

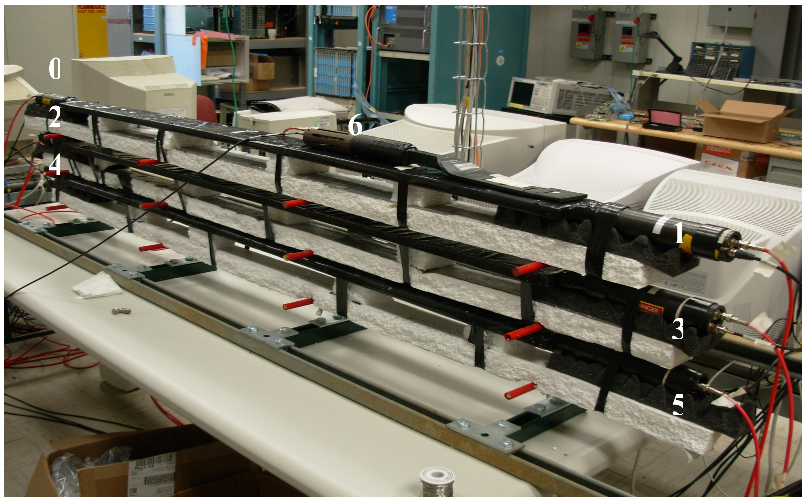

Figure 2: Photo of scintillator test stand

- *Trigger Logic*

  For the trigger logic we used physical modules in the NIM crate for most of the logic. When connecting from module to module you must use the same length cables. If you do not, the time delay will cause errors in the bin readout of the TDCs.
  - The cables from the PMTs are first run through a discriminator (PS 706) at threshold 50 mV. The small scintillator is run through a PS 706 discriminator as well.
  - The signals from the scintillators that are being tested are input into a LC 429A for 'OR' logic for each paddle to guarantee a hit.
  - The second output from the discriminator is delayed and sent to a TDC, a LC 2228A.
  - The output cables from the 'OR' and the small scintillator are put into a PS 754 as an 'AND'. This requires all paddles including the small scintillator to be hit.
  - From the PS 754, the output is run into a LC 429A to fan out the signal.
    - One signal is sent to the VME start and one is sent to a TDC.

  This is a very restrictive trigger that greatly reduces the number of events seen however, using this as a trigger reduces the number of unusable events. For the start trigger to occur before the delayed cables from the discriminator you must verify that the delay cable has a longer delay time than the cables running through the logic modules. The internal delay of each module can be looked up easily online and all cables are printed with their delay. The logic setup for the modules in the NIM crate are seen is Figure 3.

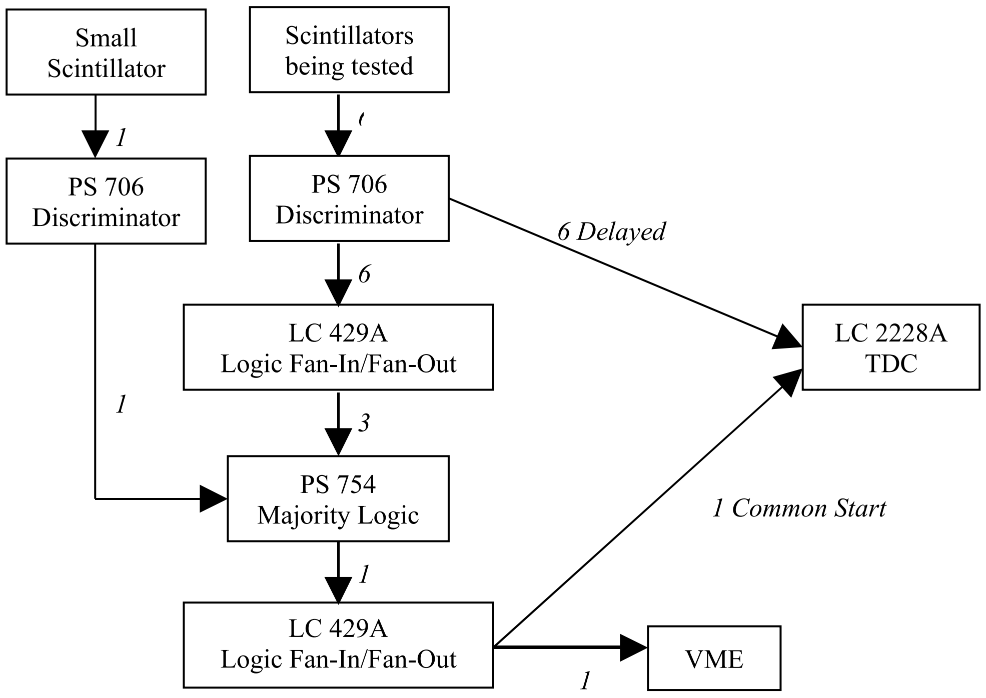

Figure 3: Basic trigger logic with small scintillator

2.6 Computer Interface

- *Connection*

  Both the VME crate and the computer are plugged into a hub for Ethernet cords.
  Computer IP: 192.168.12.1
  VME IP: 192.168.12.10
- *Initialization from computer*

  Every time you reboot the computer you must run "*/home/pdaq/server_setup.sh*" to initialize the system. For more detailed instructions see "*PDAQ: A data acquisition system for lab tests of the PHENIX detectors*," a pDAQ instruction manual (hardcopy in the lab or online).
- *Connecting from computer*

  Type the command "*ct*" in the terminal (this is an alias for "*telnet 192.168.12.10*"). This will bring up the command line to control the VME.
- *VME/CAMAC setup*

  There is a setup file on the Linux box that initializes and defines modules and settings. It is "*/home/pdaq/vxWorks/startup/startup*". The VME loads settings from this file every time it is rebooted. You can make changes in this file and reboot the VME to put them into effect, or just execute the changes in the command interface. For more details on these commands and settings see the pDAQ instruction manual (hard copy).
- *Controlling the VME*

  The command interface with the VME has a few basic commands to change settings and execute data runs. A more extensive listing is in the pDAQ instruction manual but here a few basic operation commands:

  daq_open - initializes a connection to allow a data stream between the computer and the VME
  daq_begin *n* – begins a data run that will be saved as run #n
  daq_set_max_volume - sets a max volume at which a data run will automatically end
  daq_set_max_events - sets an event limit for automatically ending.
  daq_end - ends run and writes the data file to the Linux box
  Status - gives information about the settings of an active run and events taken
  reboot - reboots the VME. (will put into effect any changes made to the setup file.)

- *Important files and directories*

  More is in the next section, but as it is concerned here, data files are actually written in "*/home/pdaq/rpc/*" in the format "*run_0000x_0y.evt*". (x is the run number and y is the run sub-number)

3. Software

This section will discuss the software used to process data and brief descriptions of how to operate it.

The framework currently used is analyzing the data using pmonitor. pmonitor is used by C code that compiles into a library which can be used in root to run analysis and work with the data real-time as it is being processed. In root the library is loaded and a set of pmonitor commands are used to load a data file and begin processing data at which point you can display or minipulate the data as needed using normal root functions while the processed data is continually updated.

- *pmonitor commands:*

  These are some of the basic commands to operate the pmonitor program in root:
  - first you must load the compiled library with:
    - root [0] gSystem->Load("libp_sc1.so");

  - poncsopen("<*data directory*>/run_0000n_0n.evt") - this opens the data file you want to process
  - **pstart( )** - begins processing the data and lets you use the root command line in the mean time
  - **pstatus( )** - give status information on actively running data
  - **pstop( ) -** stops processing data
  - **pclose( )** - closes a data file that is currently open to let you open another
  - if you have any other questions, please contact Martin Purschke

With some of the commands described, I'll explain how we implement our specific analysis into the pmonitor code. To make basic modifications, you will change this file "*/home/pdaq/danielj/pmon/p_sc1.cc*". There are two main functions in this file; *init()* and *Process_event()*. All variable to be used should be declared globally and initialized in *init(),* which is run at the execution of the **pstart()** command. Calculations, and data processing go in *Process_event()*.

Process_event() is called repeatedly for each event. You can use a condition requiring event type '9' for any functions to be executed once at the beginning of the processing and event type '12' for anything to be executed at the end of data processing. For normal events, the only required condition is that the data packets were successfully acquired for this event. From here you can fill histograms or count data as needed.

This is the basic format of the code, for more detailed questions about the pmonitor framework, contact Martin Purschke. For more questions on our initial implementation for the RPC factory, contact Daniel Jumper (dxj05a@acu.edu) .

- *Important files and directories*
  - "*/home/pdaq/rpc/*" - this is where data is written from the VME in .evt files. this will be referenced when telling the analysis program where to find data.
  - "*/home/pdaq/danielj/pmon/*" this directory contains the currently used data processing code.
    1. "*libp_sc1.so*" - the code compiles into this library to be loaded in root
    2. "*p_sc1.cc*" - this code is the meat of the analysis. Histograms are made and filled. The data is processed. Calculations are made. make changes here according to the analysis you are doing.

3. "*p_sc1_analyze_functions.cc*" - contains some simple functions used in processing the data in *p_sc1.cc*
4. "*p_sc1_analyze_functions.h" –* contains the timing calibration definitions for each PMT channel.
5. "*p_sc1Makefile*" - use the command "make -f p_sc1Makefile" to compile the code into libp_sc1.so when you make changes.
6. *"/home/pdaq/yangrz/daq-test/"* and *"/home/pdaq/danielj/sctest/"* contain earlier examples of code to fill ntuples in a root file from .evt files and analyze results but this method is no longer in use.

4. Example runs

4.1 *Example 1*

A basic run without a small scintillator as a trigger is seen here. The trigger in this setup is simply from running a series of logic modules on the three scintillators. The logic setup is as follows in Figure 4. Figure 4 is color coded for viewing ease. The red lines indicate cables from scintillator 1, black indicates cables from scintillator 2, blue indicates cables from scintillator 3, and green indicates the trigger logic.

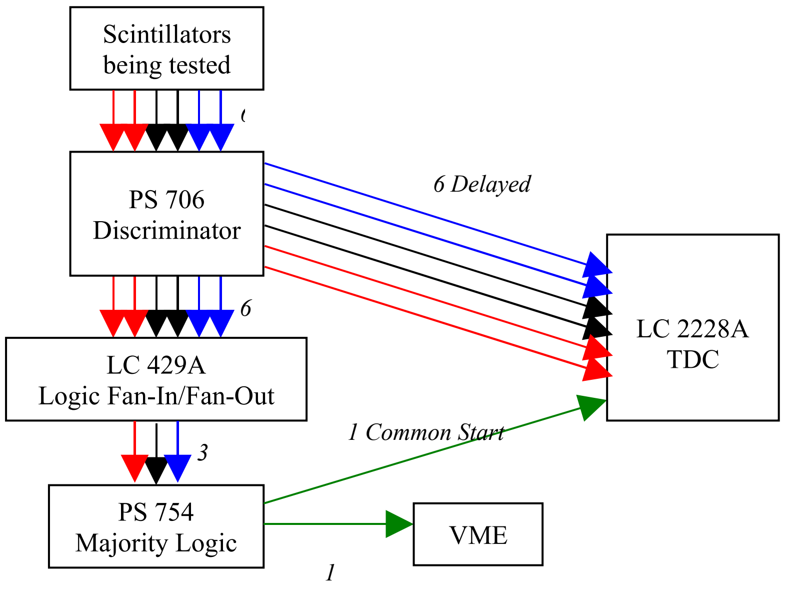

Figure 4: Basic trigger logic

Cables run from each PMT on each scintillator paddle. The cables run through a PS 706 discriminator. There are two outputs from the discriminator, one set is delayed and sent to a TDC, the other set is sent into a LC 429A logic module where each paddle's cables are put through an 'OR.' The output of the LC 429A is now three cables, one for each scintillator paddle. The three outputs are put into a PS 754 where a 2/3 majority was implemented. The output of the PS 754 acts as the start for the VME and the TDC. Data taken from three runs setup in this manner are seen in Table 1.

Table 1

Channel	*1*	*2*	*3*	*4*	*5*	*6*
Run A Bin #	425	424	421	404	416	413
Run B Bin #	424	423	420	403	415	412
Run C Bin #	420	425	418	402	414	411

4.2 *Example 2*

In this run we implemented a small scintillator for use as a trigger. The color-coding from above is still the same with the addition of purple as the signal cable from the small scintillator. The logic setup is seen in Figure 5.

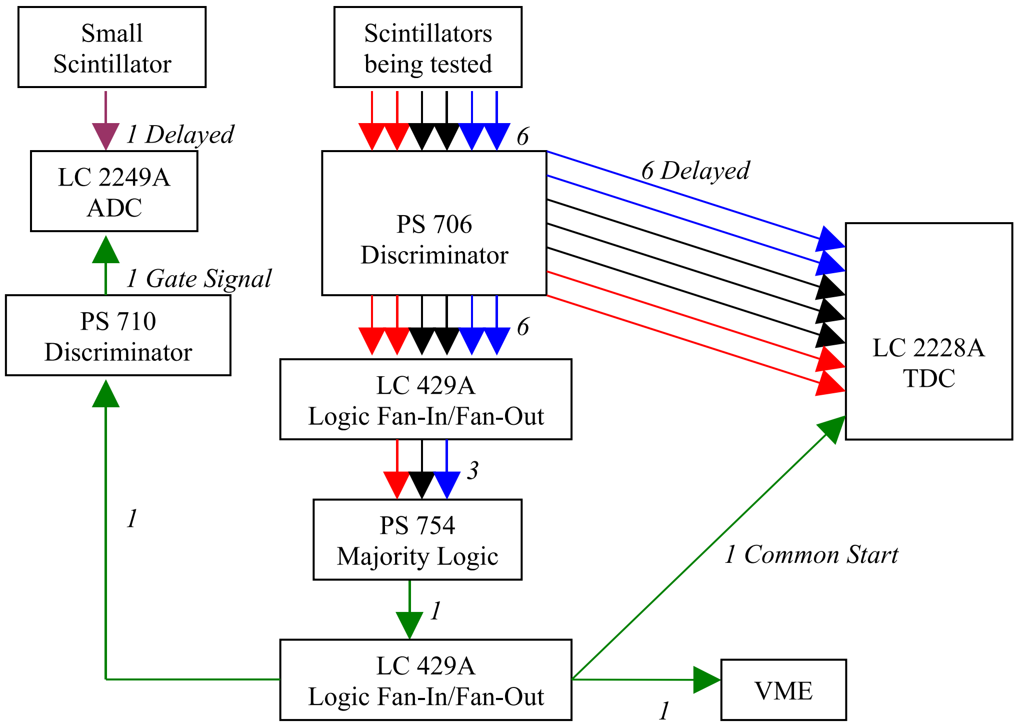

Figure 5: Logic setup using a small scintillator and an ADC

By using a small scintillator that is known to be functional as part of the trigger, we are guaranteeing a hit on that portion of the scintillator. The logic for the common start and the

VME start are the same as above. The use of an ADC for the small scintillator allows us to only examine events that occur in a small time period (a gate). To set the gate width for the ADC we send it a signal from a discriminator where we can manually adjust the pulse width with a small screwdriver. The signal from the discriminator is the same signal used as the common start. Using this logic we guarantee the 2/3 majority hits on the large paddles and a hit on the small scintillator. This trigger logic is much stricter than those used previously. Sample data and offset calculations from runs taken using this setup are seen in Table 2.

Table 2

	length (cm)	bins
Between sc1 and sc2	13.97	9.3133
Between sc2 and sc3	13.33	8.8867

run_00003_01.evt
~2,200,000 events

Channel	0	1	2	3	4	5	427	sc1
Average	471	542	341	372	357	537	437	sc2 // average
StDev	115	120	89	126	39	115	446	sc3
Total Offset	*44*	*115*	*-96*	*-65*	*-89*	*91*		

run_00004_01.evt
~2,200,000 events

Channel	0	1	2	3	4	5	428	sc1
Average	467	546	346	371	357	537	437	sc2 // average
StDev	115	124	93	124	44	116	446	sc3
Total Offset	*39*	*118*	*-91*	*-66*	*-89*	*91*		

Average Offset	***41***	***116***	***-94***	***-66***	***-89***	***91***

Appendix

"PDAQ: A data acquisition system for the lab tests of the PHENIX detectors:"
www.rhic.bnl.gov/~purschke/pdaq.ps

DAQ modules used for scintillator efficiency testing:

LOGIC

Model 429A Quad Mixed Logic Fan-In/Fan-Out
Company: LeCroy
http://www-esd.fnal.gov/esd/catalog/main/lcrynim/429a-spec.htm

- Combines the operations of TTL-to-NIM level translation, logic fan-in, logic fan-out, and polarity inversion in one module.
- Each of the four channels has four inputs which accept both NIM and TTL levels.
- Each channel of the Model 429A contains four independent logic inputs, four normal logic outputs, and two complementary logic outputs.
- Channels may be paralleled to provide up to 16 inputs and 24 outputs by means of a front-panel switch.
- Inputs are 50 Ohm impedance for NIM or TTL signals.
- Unused inputs need not be terminated.
- Inputs may be driven with single or double amplitude NIM signals or TTL signals without affecting output amplitude.
- The three pairs of bridged outputs are direct-coupled current sources which deliver -32 mA into two 50 Ohm loads.
    - Output duration is equal to the logical sum of the input durations.
- The circuitry of the Model 429A is complete direct-coupled and compatible with either normal or complementary logic signals in any duty ratio.

Model 754 Four Input Majority Logic Unit
Company: Phillips Scientific
http://www.phillipsscientific.com/preview/754pre.htm

- Contains four channels of four input logic with veto in a single width NIM module.
- Logical AND, OR, Majority logic, Fan-in/Fan-out, and Anti-coincidence functions
- Functions are direct coupled and operate to over 300 MHz with input overlap times as narrow as 750pSec.
- Each channel has four logic inputs, an anti-coincidence input, a coincidence level switch, and five outputs with common width control.
- The inputs are enabled by connecting the input cable to the desired input, eliminating errors often occurring with switched inputs.

- The setting of the coincidence level switch then determines whether a logical OR, AND, or Majority logic function will produce an output.

DISCRIMINATORS

Model 710 Leading Edge Discriminator
Company: Phillips Scientific
http://www.phillipsscientific.com/preview/710pre.htm

- Eight channel, leading edge discriminator packaged in a single-width NIM module.
- Independent threshold and width controls, a fast veto for inhibiting, a prompt linear summed output, and a versatile output configuration with four updating outputs per channel.
- Up to 16 channels can be "OR'D" directly by cable to other summed outputs providing a versatile scheme to form a trigger.
- A front panel LEMO input accepts a NIM level pulse for fast simultaneous inhibiting of all eight channels.
- Secondly, a slow bin gate via the rear panel connector inhibits the module and is enabled or disabled from a rear panel slide switch.
- The updating design permits deadtimeless operation which is desirable for fast coincidence applications at high rates.
- The 710 has four high-impedance current switching outputs per channel.
    - They are configured as one pair of double amplitude bridged outputs, one normal NIM level and one complemented NIM level.

Model 706 Leading Edge Discriminator
Company: Phillips Scientific
http://www.phillipsscientific.com/preview/706pre.htm

- 16 Channels in Single Width NIM Module
- 100 MHz Input to Output Rate
- Common Threshold Control -10 mV to -1 Volt
- Common Width Control 5 nS to 150 nS
- Fast Common Veto and Bin Gate
- One Pair Bridged Outputs per Channel
- Reliable Current-Switched Outputs

HIGH VOLTAGE

Model 4032A High Voltage System
Company: LeCroy

http://www.lecroy.com/lrs/dsheets/dslib.htm

- 32 high voltage outputs
- negative voltage

TDCs

Model 2228A Octal Time Digitizers
Company: LeCroy
http://www.lecroy.com/lrs/dsheets/2228.htm

- 11 bit (timeout at 2048 bins)
- 3 bin settings; 50 ps 100 ps, 250 ps

ADCs

Model 2249A Charge Analog-To-Digital Converter
Company: LeCroy
http://www.lecroy.com/lrs/dsheets/2249.htm

- 12 channel
- 10 bit resolution
- Input sensitivity of the Model 2249A is 0.25 pC/count for a full scale range of 256 pC.
    - This is compatible with most available signal sources and no additional buffering or reshaping of any kind is required to digitize nanosecond pulses.
- Able to begin digitizing on the command of a prompt gate and be reset, if necessary, before the end of conversion on the basis of delayed logic or chamber information.